Grammatical Inference as Class Discrimination

Menno van Zaanen and Tanja Gaustad

TiCC, Tilburg University
Tilburg, The Netherlands
{M.M.vanZaanen, T.Gaustad}@uvt.nl

Abstract. Grammatical inference is typically defined as the task of finding a compact representation of a language given a subset of sample sequences from that language. Many different aspects, paradigms and settings can be investigated, leading to different proofs of language learnability or practical systems. The general problem can be seen as a one class classification or discrimination task. In this paper, we take a slightly different view on the task of grammatical inference. Instead of learning a full description of the language, we aim to learn a representation of the boundary of the language. Effectively, when this boundary is known, we can use it to decide whether a sequence is a member of the language or not. An extension of this approach allows us to decide on membership of sequences over a collection of (mutually exclusive) languages. We will also propose a systematic approach that learns language boundaries based on subsequences from the sample sequences and show its effectiveness on a practical problem of music classification. It turns out that this approach is indeed viable.

Keywords: empirical grammatical inference, class discrimination, *tf*idf*

1 Introduction

Grammatical inference deals with the learning of languages. The task is typically defined as follows: Given a set of example sequences, find a compact representation of the underlying language of which the sequences are examples. The compact representation is called a grammar, the example sequences are generated from the grammar by a teacher and it is the learner that aims to find the underlying grammar.

The field of grammatical inference is often divided into two subfields: formal and empirical grammatical inference [1]. Formal grammatical inference investigates learnability of classes of languages given a particular learning setting. The result of this research is a formal, mathematical proof showing that a certain class or family of languages is learnable (or not) provided the environment corresponds to the requirements of the learning setting. Probably the most famous of these settings is that of identification in the limit [2], but others exist [3].

Here, however, we are more interested in empirical grammatical inference. In contrast to formal grammatical inference, where mathematical proofs are provided on learnability of predetermined classes of languages, empirical grammatical inference deals with learning of languages in situations where the underlying

grammar or class of grammars is not known. This typically leads to empirical results on naturally occurring data and addresses practical learning situations.

In the ideal case, we would like to combine both formal and practical grammatical inference techniques. This means that we know formally that languages can be learned in the setting under consideration and that in practice this is also true. Knowing that a language is learnable formally does not necessarily mean that it is also learnable in practice, due to, for instance, noise, limited amounts of available data, or a (minor) mismatch between the practical and formal learning settings.

In this paper, we propose to treat the problem of empirical grammatical inference in a slightly different way. Instead of trying to learn a full, compact representation of the underlying language, we redefine the task to find a representation of the *boundary* of the language. In many cases, both the learned grammar or the learned boundaries can be applied. For instance, when the learned construction is used to classify sequences into classes (such as inside or outside the language), both representations are equally applicable.

In addition to the theoretical specification of our new language learning approach, we describe a practical implementation of the approach. This implementation relies on finding patterns in the shape of subsequences from the example sequences for each of the languages. (In the case of learning sequence membership of one language, negative examples are considered as an alternative language.) The patterns that have high predictive power are selected and are subsequently used to classify new sequences.

The paper is structured as follows. Firstly, we will specify the new approach to empirical grammatical inference in more detail, including a discussion of the advantages and disadvantages as well as a description of a practical system. Next, the results of applying the practical system to two data sets are provided. The paper ends with a conclusion.

2 Approach

The research presented in this paper introduces two novelties. First, we redefine the task of grammatical inference as a discrimination task. The new task is to identify the boundary of the underlying language(s) rather than to construct a compact representation of it (in the form of e.g. a grammar). Second, we propose a practical system that identifies patterns that describe language boundaries based on the example sequences. We apply an existing statistical measure to identify patterns that are useful for the identification of the boundary. Both aspects will now be described in more detail.

2.1 Class Discrimination

Languages can be visualized in the sequence space (the space that contains all possible sequences) and are typically described as an area in the shape of a circle or oval, like a Venn diagram. The area contains all sequences that are part of

the language and all sequences outside the area are non-members. Typically, the aim of grammatical inference is to find a grammar that describes the entire area of the language.

Most often, grammatical inference approaches aim to learn a representation in the form of a grammar that fully describes the underlying language from which the example sequences are drawn. The advantage of learning a full description is that this can also be used to generate more sequences in the language, which leads to a proper generalization of the sample sequences. However, grammatical inference settings, such as identification in the limit or PAC learning do not specify that such a full description from a generative point of view is required.

In contrast to learning a full description of the language, we propose to find a representation of the line describing the boundary of the language only. Once we know this boundary, essentially we also know which sequences are in the language and which are out, without having an explicit representation of the sequences that are part of the language (which one has in the case of a grammar). Note however, that generating sequences (in addition to the ones known from the learning sample) in the language is non-trivial in this case.

When looking for the boundary between languages (where in the case of learning one language L, the other language would be its complement L^C), we do not need to know exactly which sequence is inside the language. We are only interested in sequences that are close to the boundary of the language.

This idea of finding a representation of the boundary of the language can be compared to the supervised machine learning methods based on support vector machines (SVMs) [4]. Given a set of training examples belonging to either one of two categories, an SVM model is built that predicts into which category a new, unseen example falls. The model represents the examples as points in the instance space mapped in such a way that the examples from the two categories are divided by a clear margin. Ideally, the boundary falls right in the middle of the margin and this boundary represents the largest distance to the nearest training data points of any class, thereby minimizing the generalization error of the SVM classifier. Unseen examples are mapped into the instance space and, based on which side of the boundary they fall on, their class is predicted. Interestingly, SVMs only rely on examples that are close to the boundary. Examples that are far away from the boundary are not used to build the vector that distinguishes the areas describing the classes.

Alternatively, our approach can be seen as being similar to the k-NN (Nearest Neighbor) supervised machine learning approach [5]. Here, just as in the SVM case, the training instances are placed in the instance space. Classification of a particular (unseen) instance is then performed by finding the instances from the training data that are closest to the unseen instance. The assigned class is found by taking a majority vote over all classes of the nearest instances. The boundaries in this situation are computed on the fly.

In a way, the k-NN approach does not aim to learn a complete description of the boundary in the sense of a formula describing that boundary. Whereas SVMs aim to learn linear classifiers on a mapped instance space (allowing for

non-linear classification), k-NN only computes local boundaries when required for classification. At no point in time a complete formal description of the boundary is known (although this can be extracted from the known instances in the instance space if required).

With the approach described here, we essentially treat the task of grammatical inference as a discrimination task. Without creating a description of all the sequences in the language, we can still decide for unseen sequences in which area of the sequence space they should be placed. It also means means that such an inference system can be used to distinguish between one or more languages at the same time. The difference there is that boundaries between each of the languages need to be learned.

Note that the practical approach we will describe here identifies patterns that can be used to distinguish language membership of example sequences. Each pattern only describes a small part of the language boundary. In that sense, it fits in between SVMs and the k-NN classifiers. The patterns are simple (just like the simple representation used in the SVM context) and each one describes a small part of the boundary, just like a k-NN classifier does.

So far, we have not said anything about the properties of the boundaries. For instance, what shape the boundaries should have or whether the boundaries may overlap (allowing sequences to be in multiple languages at the same time). We will discuss some properties of the boundaries in the next section, which describes a practical system. However, more work needs to be done in this area for alternative practical systems.

2.2 *tf*idf* Pattern Identification

The discussion so far has been quite abstract. It may be unclear exactly how we should find the boundaries between languages or perhaps even how we should describe these boundaries. To show that this abstract idea can actually lead to a practical system, we will propose a working system that is entirely based on the theoretical approach that was described in the previous section.

The representation of the boundary between languages we use here consists of subsequences. These are consecutive symbols that occur in the example language sequences that the system received during learning. In fact, for practical purposes, we search for subsequences of a certain length, which means they can be seen as n-grams (with n describing the length of the subsequence).

By using n-grams as the representation of our patterns, we explicitly limit the languages we can identify. In fact, using patterns of a specific length, we can learn the boundaries of the family of k-testable languages [6]. This family contains all languages that can be described by a finite set of subsequences of length k. It may be clear that these subsequences of length k correspond well with our patterns of fixed length n.

Note, however, that we do not present a formal proof of learnability of this family of languages (which has already been shown before [7]), but we will implicitly assume that the language(s) we are trying to learn are in fact k-testable

or if they are not, we will provide an approximation of the language that is k-testable.

The subsequences we are interested in should help us decide whether an unseen sequence is part of the language (or in the more generic case, it should help us identify which language the sequence belongs to). Therefore, we will use the subsequences as patterns. During testing, the patterns are matched against the to be classified sequence (counting number of occurrences per language). Based on this information, the sequence is classified.

For the patterns to be maximally useful, during learning we would like to identify patterns (i.e. subsequences in the shape of n-grams) that are maximally discriminative between languages and that at the same time occur often.

To measure the effectiveness and usability of the patterns, we apply a classic statistical measure from the field of information retrieval, namely the "term frequency*inverse document frequency" (*tf***idf*) weight [8]. This measure consists of two terms, term frequency (*tf*) which measures the regularity and inverse document frequency (*idf*) which measures the discriminative power of the pattern.

Originally, in the context of information retrieval, the *tf***idf* weight is used to evaluate how relevant a document in a large document collection is given a search term. In its classic application, *tf***idf* weights are computed for all documents separately in the collection with respect to a search term.

The first part of the *tf***idf* metric is *tf*. It is defined as the number of times a given term appears in a document. Simply counting the number of occurrences, will yield a bias towards longer documents. To prevent this, the *tf* measure is often normalized normalized by the length of the document. This results in the following metric:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{1}$$

where $n_{i,j}$ describes the number of occurrences of term t_i in document d_j. The denominator represents the length of document d_j, which is measured as the total number of terms in document d_j.

The idea behind *tf* is that when the term t_i occurs frequently in certain documents, these documents are considered more relevant to the term than documents with fewer instances. Taking this into the extreme, when no occurrences of the term are found in a document that document is probably not about the topic represented by the term. (In the case of natural language terms, this may not always be true. In fact, this has led to research into, for instance, stemming, pseudo relevance feedback and automatic synonym generation [9].)

The second part of the *tf***idf* is *idf*. For a given term t_i, it is calculated as follows:

$$idf_i = \log \frac{|D|}{|\{d : t_i \in d\}|} \tag{2}$$

where $|D|$ is the total number of documents in the collection and $|\{d : t_i \in d\}|$ is the number of documents that contain the term t_i.

The *idf* measures relevance of a term with respect to the documents. Intuitively, this can be described as follows. On the one hand, terms that occur in all documents are not particularly useful when deciding which document is relevant. On the hand, terms that occur only in one or a few documents are good indicators, as those documents are probably about the term under consideration.

To obtain the *tf*idf* weight for a particular term, the term frequency *tf* and inverse document frequency *idf* are combined:

$$tf{*}idf_{i,j} = tf_{i,j} \times idf_i \tag{3}$$

The default way of computation of *tf*idf* provides us with an indication of how relevant a particular document is to a particular term. This metric can be extended, resulting in *tf*idf* scores for multiple terms. In this case, the *tf*idf* for all documents is computed for each of the terms. These *tf*idf* values are then summed and the documents that have the highest *tf*idf* scores (representing that these documents are most relevant with respect to the terms) are preferred.

In the research presented here, we extend the *tf*idf* metric in a different way. Instead of computing the *tf*idf* score of a collection of terms (in the sense of a "bag-of-terms"), we want to be able to compute the *tf*idf* score of a sequence of terms with a fixed order. This corresponds to treating n-grams (a sequence of terms) as if it is a single term. The underlying idea behind using sequences of terms instead of single terms is that we think that sequences are more informative than single terms to determine the boundary between languages (and this will be shown empirically in Section 3).

The modification of the computation of the *tf*idf* weights is rather straightforward. Instead of counting single terms (for instance in the computation of the *tf*), n-grams are counted as if they are single terms (with single terms being a specific case where $n = 1$). For instance, $n_{i,j}$ is the number of occurrences of a particular n-gram t_i in document d_j.

To summarize, during the learning phase, the learner receives example sequences from the languages under consideration. Out of these sequences, all n-gram patterns are extracted and for each of these, the *tf*idf* score is computed (with respect to each of the languages). Patterns that have a non-zero *tf*idf* are retained as patterns for classification afterwards. Note that if patterns occur in all languages, their *idf* will be zero (and the *idf* will be high if it only occurs in one language). At the same time, if the patterns occur more often, they are considered more important, which increases the overall *tf*idf* value for that pattern due to a higher *tf*.

During classification, a new, unseen sequence is presented. All patterns are matched against it, leading to a score for each of the languages. This score is calculated by summing the *tf*idf* scores for each match of a pattern, keeping track of the *tf*idf* per language. The sequence is then classified into the language that has the highest combined *tf*idf* value.

In Section 3 we will describe experiments performed with fixed length n-grams, but also with n-grams of varying sizes. This brings up an interesting aspect of *tf*idf*. Shorter patterns (with small n) have a higher likelihood of

occurring compared to longer patterns (with large n). This means that the *tf*idf* will typically be higher for short patterns. To reduce this effect, we multiply each *tf*idf* score by n, the length of the n-gram. This leads to a higher impact for longer patterns (which, if they can be found in the sequence to be classified, gives more pronounced evidence that the sequence actually belongs to that language).

2.3 Imperfect Languages and Noise

So far, we have assumed that there is a perfect distinction between the languages. In the simplest case, we consider a language L and its complement L^C. This means that all possible sequences come from either L or L^C.

In practice, it might be that the situation is more difficult. Firstly, there may be an area in sequence space that is not described by any language. This happens when the sequence space is not perfectly partitioned. In other words, the sequence space S is not entirely covered by the languages $(L_1, \ldots, L_n)$: $S \supset \bigcup_{i=1}^{n} L_i$. In this case, sequences exist that are not a member of any language. The system will decide (perhaps randomly) that the sequence is a member of one of the known languages, because it assumes that the entire sequence space is covered by the languages.

Secondly, there may be an overlap between the languages. For instance, sequences that really belong to L are presented to the learner as sequences from L^C or vice versa. If this occurs, the training data contains noise.

A major advantage of the use of *tf*idf* in this system is that if noise occurs in the data, the patterns dealing with the subsequences containing the noise are now automatically ignored in the pattern identification phase. This works through the *idf* component in the *tf*idf* formula. When noise introduces sequences in the wrong language, the patterns that would otherwise have been found (because they are distinctive for the sequences in a particular language) will now receive a zero *idf* and hence a zero *tf*idf*, which then results in the pattern being dropped. This allows for a very robust practical system.

3 Empirical Results

To empirically evaluate the effectiveness of the *tf*idf* pattern identification and discrimination approach to detecting boundaries between languages, we test this approach in two practical experiments. The next section describes the data sets and classification tasks used, followed by an explanation of the data representation.

3.1 Data Sets and Classification Tasks

To evaluate our approach, we compiled two separate data sets from the area of music classification. Both data sets were retrieved from the **kern scores website[1] [10].

[1] http://kern.ccarh.org/

The two different data sets lead to two different classification tasks. Firstly, we have a binary class data set containing folksongs. One class (i.e. language) consists of Asian folksongs and the other of European folksongs. Both are taken from the Essen Folksong Collection. This data set is called *country*. An overview of the data set can be found in Table 1.

We will use these data sets to show the feasibility of the approach. Music has a fairly limited amount of symbols (compared to for instance natural language), but the training data is extracted from real world data. Music also has inherent "rules" or restrictions, which we aim to learn here. Furthermore, music allows us to experiment with different representations easily.

Table 1. Overview of the country data set

Class	Description	# of pieces
Asia	Chinese folksongs (4 provinces)	2,241
Europe	European folksongs (19 countries plus misc)	848
Total		3,089

The aim of the country classification task is discriminating folksongs. Two classes are distinguished: Asian folksongs and European folksongs. Even though the original data set has more fine-grained classes, we have not tried to further distinguish either collection into sub-classes (e.g. different provinces or countries) as we expect there to be a partial overlap between the songs from different European countries.

Intuitively, the country task is relatively easy for several reasons. There are only two classes to classify into (compared to four in the other task). Also, we expect that the difference between Asian and European folksongs will be quite pronounced. However, the musical pieces to be classified are relatively short, which might make identifying and matching patterns, and hence classification, more difficult.

Secondly, we have have extracted the musedata selection from the **kern scores website, which contains pieces by four composers: J.S. Bach, A. Corelli, J. Haydn, and W.A. Mozart. We call this data set *composer* and numerical information on the data set is shown in Table 2.

In the composer classification task, the system should identify which composer, out of the four composers, composed a given musical piece. The system selects one out of four classes (Bach, Corelli, Haydn, and Mozart). Note that the composers come from different, but overlapping periods.

One has to keep in mind that the composer classification task is actually quite difficult. For instance, when people are asked to distinguish between musical pieces from these composers (see e.g. the "Haydn/Mozart String Quartet

Table 2. Overview of the composer data set

Class	Description	# of pieces
Bach	chorales and various	246
Corelli	trio sonatas	247
Haydn	quartets	212
Mozart	quartets	82
Total		787

Quiz"[2]), the identification accuracies are only 55% and 57% for Mozart and Haydn respectively. Given these results, we expect this task to be hard for automatic classification as well.

3.2 Data Representation or Features and Patterns

We start with the collections of musical pieces in the humdrum **kern format [11]. This format is a symbolic representation of sheet music. Because we want to identify patterns in the musical pieces, we need to define exactly which aspects of the musical representations are going to be used to define the patterns. We convert the music from the **kern humdrum format to a simpler format describing melody (pitch) and rhythm (duration) only. This information is extracted directly from the humdrum **kern format and converted into a new symbolic representation.

For both pitch and duration, we chose one way of rendering, namely using what is typically called *absolute* representations. *Absolute pitch* refers to the absolute value (in semitones) of the melody with $c = 0$ (e.g. $d = 2$, $e = 4$, etc.). Similarly, *absolute duration* gives the absolute duration of a given note (e.g. 2, 16). This absolute representation allows for a one-to-one mapping from the **kern humdrum representation of sheet music to a simple symbolic representation that can be used to learn.

We know that alternative representations of symbolic music are possible [12, 13] and will perhaps even lead to better results. However, here we have selected a fairly simple representation, which allows us to demonstrate the feasibility of the new language learning approach.

To make the meaning of the n-gram patterns explicit: the patterns with $n = 1$ correspond to patterns of a single note in a piece of music. When $n = 2$, the patterns describes two consecutive notes, etc. Other representations of the music may lead to patterns that describe more complex aspects of music (potentially non-consecutive notes or more abstract descriptions of the music).

Each piece of music is converted to a sequence of symbols, where each symbol is a combination of the pitch and duration of a single note. This means that each symbol in the representation that is used to find patterns consists of two

[2] http://qq.themefinder.org/

components (pitch and duration) that are "glued" together, leading to a single symbol.

Starting from the converted sequences of symbols for each of the musical pieces, we combined them into classes. Each class contains all the sequences (i.e. musical pieces) of a single composer or geographical area. These collections of sequences are used as input from which we build various patterns of n-grams as outlined in Section 2.

We assume that each composer or geographical area has its own "language" which was used to generate musical pieces. The task is then to learn the boundaries between the languages, which allows us to classify new, unseen musical pieces into the corresponding classes. (Unfortunately, this approach does not explicitly allow us to generate new music that is similar to existing musical pieces of a particular language or class.)

With respect to the shape of the patterns, we tried n-grams of size $n = 1, \ldots, 7$ and also tried combinations of n-grams of length $1-2, \ldots, 1-7$. The experiments based on the combinations of n-grams use patterns of n-grams of all the specified lengths combined. Remember that the *tf*idf* score is multiplied by the length of the n-gram, which means that longer patterns will have more impact in the final score.

The main disadvantage of the current music representation is that only local patterns can be found. For instance, languages that require global information in a pattern (such as the number of symbols in the sequence) simply cannot be identified with the current system using n-grams. This problem might be solved if a more complex representation of the data or a completely different shape of patterns is used. The solution to this problem should, however, be seen as future work.

3.3 Quantitative Results

Table 3 contains the results of applying the *tf*idf* grammatical inference pattern finding system to the two data sets. The figures describe accuracy (% of correctly classified musical pieces divided by the total number of classified pieces), combined with the standard deviation (in brackets). All experiments are performed using ten fold cross-validation.

The results clearly show that using *tf*idf* to identify useful patterns works well for both discriminating between two classes (or languages) and multiple classes (four in our case).

The first figures in the table are majority class baselines. The class occurring most often in the training data is selected and used to classify all test sequences. In the country classification, the Asian class clearly has more pieces (the accuracy is higher than the 50% that is expected with a perfectly balanced data set), whereas in the composer task, the number of instances is more balanced (expected baseline with a perfectly balanced data set would be 25%).

Looking at the results of the single size n-grams (the first seven entries following the baseline), we see that the results peak around $n = 3$ or $n = 4$. This illustrates that, on the one hand, small patterns, even though occurring

Table 3. Classification results in % correct (and standard deviation) for the country and composer classification tasks.

n-gram size	**Country classification**	**Composer classification**
Baseline	73.49 (±1.64)	27.96 (±4.01)
1	62.05 (±1.52)	64.19 (±6.79)
2	87.90 (±2.08)	78.65 (±2.25)
3	95.52 (±1.06)	81.95 (±2.85)
4	95.54 (±1.72)	79.79 (±4.31)
5	94.12 (±2.65)	78.01 (±4.01)
6	91.97 (±2.96)	74.58 (±4.84)
7	90.65 (±2.75)	71.91 (±4.57)
$1-2$	79.82 (±3.02)	76.75 (±3.93)
$1-3$	89.33 (±2.84)	81.06 (±3.31)
$1-4$	92.27 (±1.94)	81.82 (±3.56)
$1-5$	93.00 (±1.54)	82.07 (±4.25)
$1-6$	93.13 (±1.48)	81.56 (±3.91)
$1-7$	93.16 (±1.44)	81.06 (±3.77)

frequently, have less discriminative power to classify sequences in classes compared to larger n-gram patterns. On the other hand, large n-gram patterns have high discriminative power, but do not occur enough (and hence are less usable). Hence, the optimum size of the patterns is around length three or four.

The story is different when a collection of patterns of varying length is collected and used for classification. The results on the country task are still increasing after $n = 1-7$, but so far the results are worse than the best single n-gram pattern ($n = 4$). On the composer task, the results of the combination of n-gram patterns peaks at $n = 1-5$. It results in the best score for that task. However, the difference in results comparing $n = 1-5$ against $n = 3$ is not statistically significant.

Overall, the results show that the *tf*idf* pattern finding system significantly outperforms the majority class baseline. The experiments also show that there seems to be an optimum pattern length regardless of the experiment. This can be explained by considering how the *tf*idf* metric works.

4 Conclusion

Empirical grammatical inference is typically defined as the task of finding a compact representation (in the shape of a grammar) of a language, given a set of example sequences. Typically, the learned grammar is a full description of the language, often allowing for the generation of additional sequences in the language. The underlying grammar from which the example sequences are generated is often unknown, which means that evaluation of the effectiveness of the empirical grammatical inference system needs to be performed according to the classification of unseen sequences.

Here, we modified the task slightly. Instead of finding an explicit grammar for the language, we aim to find a representation of the boundary of the language. Once this boundary is known, it can be used to indicate which sequences should be considered as a member of the language or not. Generation of additional sequences is not directly supported by this representation.

The advantage of this view on empirical grammatical inference is that the system can be used to distinguish between one or more languages at the same time. Effectively, the task of grammatical inference is treated as a discrimination task. The situation that is normally seen as the grammatical inference task (learning a representation of one language) can be seen as a one-class discrimination task. However, the view that is proposed in this paper also allows for the learning of multiple languages simultaneously.

The patterns that are learned using this approach used together describe the boundary between languages. Each pattern only describes a small part of the completely boundary. Often, when classifying, only a limited amount of patterns is used to decide which language the sequence belongs to.

In addition to the new approach to grammatical inference, we have also proposed a practical system that finds patterns in example sequences. These patterns allow for the classification of new and unseen sequences into languages. Using an extension of the *tf*idf* metric, the system identifies patterns that both occur often and are helpful in discriminating the sequences. Another advantage of the presented system is that if noise occurs in the data, these sequences are automatically ignored in the pattern identification phase. This allows for a very robust system.

Applying the system to real world data sets yields good results. Two classification tasks (dividing musical data based on geography or era) have been used as experimental cases. Alternative representations of the music may still lead to improvements over the results discussed here, but these experimental results already show that this approach is practically viable.

To fully appreciate the effectiveness of the proposed approach, more experiments need to be performed. Not only should the effectiveness of different representations of the data be investigated, but completely different data sets taken from other domains should be used as well. Furthermore, to get a better idea about the state-of-the-art, the approach should be compared against other grammatical inference systems.

The main disadvantage of the current system is that only local patterns can be found. As such, languages for which global information of a sequence (such as the number of symbols in the sequence) is required, cannot be learned with the current system. This problem might be solved using a different, more complex representation of the data or, alternatively, using a completely different type of patterns. This different representation of patterns should then extend the current n-gram patterns and allow for the description of more global information. We consider this problem as future work.

References

1. Adriaans, P.W., van Zaanen, M.M.: Computational grammatical inference. In Holmes, D.E., Jain, L.C., eds.: Innovations in Machine Learning. Volume 194 of Studies in Fuzziness and Soft Computing. Springer-Verlag, Berlin Heidelberg, Germany (2006) To be published. ISBN: 3-540-30609-9.
2. Gold, E.M.: Language identification in the limit. Information and Control **10** (1967) 447–474
3. de la Higuera, C.: Grammatical inference: learning automata and grammars. Cambridge University Press, Cambridge, UK (2010)
4. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and other kernel-based learning methods. Cambridge University Press, Cambridge, UK (2000)
5. Daelemans, W., van den Bosch, A.: Memory-Based Language Processing. Cambridge University Press, Cambridge, UK (2005)
6. Garcia, P., Vidal, E.: Inference of k-testable languages in the strict sense and application to syntactic pattern recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **12** (1990) 920–925
7. Garcia, P., Vidal, E., Oncina, J.: Learning locally testable languages in the strict sense. In: Proceedings of the Workshop on Algorithmic Learning Theory, Japanese Society for Artificial Intelligence (1990) 325–338
8. van Rijsbergen, C.J.: Information Retrieval. 2nd edn. University of Glasgow, Glasgow, UK (1979) Printout.
9. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley Publishing Company, Reading:MA, USA (1999)
10. Sapp, C.S.: Online database of scores in the humdrum file format. In: Proceedings of the sixth International Conference on Music Information Retrieval (ISMIR); London, United Kingdom. (September 2005) 664–665
11. Huron, D.: Humdrum and kern: selective feature encoding. In Selfridge-Field, E., ed.: Beyond MIDI: The handbook of musical codes. Massachusetts Institute of Technology Press, Cambridge:MA, USA and London, UK (1997) 375–401
12. Conklin, D., Anagnostopoulou, C.: Representation and discovery of multiple viewpoint patterns. In: Proceedings of the 2001 International Computer Music Conference, International Computer Music Association (2001) 479–485
13. Geertzen, J., van Zaanen, M.: Composer classification using grammatical inference. In: Proceedings of the MML 2008 International Workshop on Machine Learning and Music held in conjunction with ICML/COLT/UAI 2008, Helsinki, Finland. (2008) 17–18